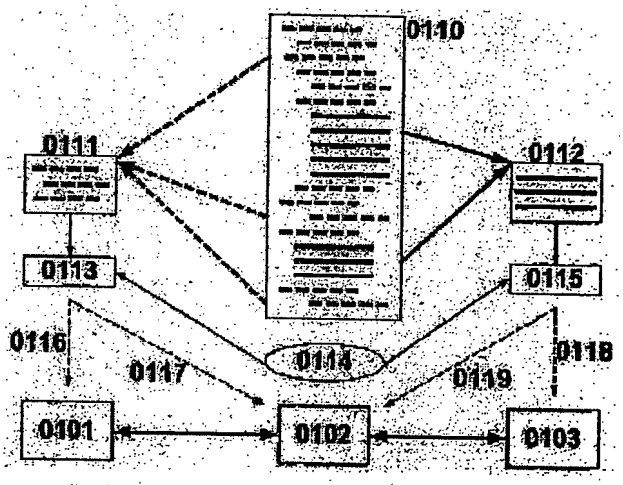
Integrated cell matrix circuit has at least 2 different types of cells with interconnection terminals positioned to allow mixing of different cell types within matrix circuit

Patent number: DE10129237
Publication date: 2002-04-18
Inventor: MAY FRANK (DE); NUECKEL ARMIN (DE); VORBACH MARTIN (DE); WEINHARDT MARKUS (DE); CARDOSO JOAO MANUEL PAIVA (PT)
Applicant: PACT INF TECH GMBH (DE)
Classification:
- international: G06F9/45
- european: G06F15/78R
Application number: DE20011029237 20010620
Priority number(s): WO2000EP10516 20001009; EP20010102674 20010207

Abstract of **DE10129237**

The circuit (1) has a number of adjacent cells (2a-2e) divided into at least 2 cell types, the cells having a sufficient size for integration of a number of logic elements (3a,3e), at least some of the cells having programmable logic elements. The terminals (5a-5d) for the interconnections between the cells are positioned to allow mixing of the different types of cells within the cell matrix. An Independent claim for a design method for an integrated cell matrix circuit is also included.



(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENT- UND MARKENAMT

(12) **Offenlegungsschrift**

(10) **DE 101 29 237 A 1**

(51) Int. Cl.[7]: G 06 F 9/45

(21) Aktenzeichen: 101 29 237.6
(22) Anmeldetag: 20. 6. 2001
(43) Offenlegungstag: 18. 4. 2002

DE 101 29 237 A 1

(30) Unionspriorität:

00/10516	09. 10. 2000	MN
00/10516	09. 10. 2000	TT
011026747	07. 02. 2001	CY

(71) Anmelder:

PACT Informationstechnologie GmbH, 80807 München, DE

(74) Vertreter:

Pietruk, C., Dipl.-Phys., Pat.-Anw., 76229 Karlsruhe

(72) Erfinder:

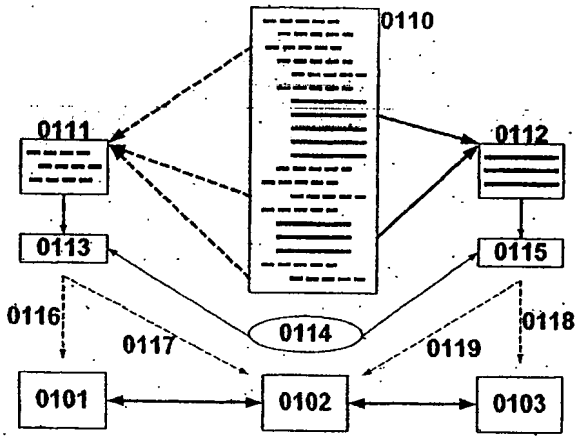
May, Frank, 81927 München, DE; Nückel, Armin, Dr., 76777 Neupotz, DE; Vorbach, Martin, 80689 München, DE; Weinhardt, Markus, Dr., 80339 München, DE; Cardoso, Joao Manuel Paiva, Vila de Paiva, PT

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

Prüfungsantrag gem. § 44 PatG ist gestellt

(54) Verfahren zur Bearbeitung von Daten

(57) Die Erfindung betrifft ein Verfahren zur Übersetzung von Programmen auf ein System, bestehend aus wenigstens einem ersten Prozessor und einer rekonfigurierbaren Einheit. Hierbei ist vorgesehen, daß die Codeteile, die für die rekonfigurierbare Einheit geeignet sind, bestimmt und extrahiert werden und der verbleibende Code zur Abarbeitung durch den ersten Prozessor derart extrahiert wird.



DE 101 29 237 A 1

Beschreibung

[0001] Die vorliegende Erfindung befaßt sich mit herkömmlichen, d. h. konventionellen und rekonfigurierbaren Architekturen sowie mit Verfahren hierfür, die eine Übersetzung einer klassischen Hochsprache (PROGRAMM), wie Pascal, C, C++, Java, etc. ermöglichen, insbesondere auf eine rekonfigurierbare Architektur.

[0002] Unter einer konventionellen Prozessorarchitektur (PROZESSOR) werden vorliegend beispielsweise sequentielle Prozessoren mit einer von-Neumann- oder Havardarchitektur verstanden, wie z. B. Kontroller, CISC-, RISC-, VLIW-, DSP- u. ä. Prozessoren.

[0003] Unter einer rekonfigurierbaren Zielarchitektur werden vorliegend Bausteine (VPU) mit konfigurierbarer Funktion und/oder Vernetzung verstanden, insbesondere integrierte Bausteine mit einer Mehrzahl von ein- oder mehrdimensional angeordneten, arithmetischen und/oder logischen und/oder analogen und/oder speichernden Baugruppen, die direkt oder durch ein Bussystem miteinander verbunden sind.

[0004] Zur Gattung dieser Bausteine zählen insbesondere systolische Arrays, neuronale Netze, Mehrprozessor-Systeme, Prozessoren mit mehreren Rechenwerken und/oder logischen Zellen, Vernetzungs- und Netzwerkbausteine wie z. B. Crossbar-Schalter, ebenso wie bekannte Bausteine der Gattung FPGA, DPGA, XPUTER, etc. Hingewiesen wird insbesondere in diesem Zusammenhang auf die folgenden Schutzrechte desselben Anmelders: P 44 16 881.0-53, DE 197 81 412.3, DE 197 81 483.2, DE 196 54 846.2-53, DE 196 54 593.5-53, DE 197 04 044.6-53, DE 198 80 129.7, DE 198 61 088.2-53, DE 199 80 312.9, PCT/DE 00/01869, DE 100 36 627.9-33, DE 100 28 397.7, DE 101 10 530.4, DE 101 11 014.6, PCT/EP 00/10516, EP 01 102 674.7. Diese sind hiermit zu Offenbarungszwekken vollumfänglich eingegliedert.

[0005] Es hat sich gezeigt, daß es bestimmte Verfahren und Programmabläufe gibt, die sich besser mit einer rekonfigurierbare Architektur abarbeiten lassen als mit einer konventionellen Prozessorarchitektur. Umgekehrt gibt es auch solche Verfahren und Programmabläufe, die besser mit einer konventionellen Prozessorarchitektur ausgeführt werden können.

[0006] Die Aufgabe dieser Erfindung besteht darin Neues für die gewerbliche Anwendung bereitzustellen.

[0007] Die Lösung dieser Aufgabe wird in unabhängiger Form beansprucht.

[0008] Es wurde erkannt, daß es wünschenswert ist, daß Verfahren zur Datenverarbeitung so ausgelegt werden, daß nur die jeweils für die rekonfigurierbare Zielarchitektur besonders geeigneten Teile des zu übersetzenden Programmes extrahiert werden. Die verbleibenden Teile des Programmes können dann auf eine konventionelle Prozessorarchitektur übersetzt werden.

[0009] Weiterhin sei angemerkt, daß die Verfahren auch auf Gruppen von mehreren Bausteinen angewendet werden können.

Systemaufbau

[0010] Ein PROZESSOR wird derart mit einer oder mehreren VPU(s) verbunden, daß ein effizienter Informationsaustausch, insbesondere in Form von Daten- und Statusinformation möglich ist.

[0011] Die Anordnung eines herkömmlichen Prozessors und eines rekonfigurierbaren Prozessors, dergestalt, daß eine Daten- und Statusinformation zwischen denselben während der Abarbeitung eines oder mehrere Programme

möglich ist und/oder ohne daß insbesondere die Datenverarbeitung auf dem rekonfigurierbaren Prozessor und/oder dem herkömmlichen Prozessor signifikant unterbrochen werden muß, sowie die Ausbildung eines derartigen Systems soweit aus dem nachfolgenden ersichtlich, wird gleichfalls beansprucht.

[0012] Es werden zunächst beispielsweise folgende Verbindungsverfahren und -mittel verwendet:

a) Shared-Memory
b) Netzwerk (beispielsweise Bussysteme wie z. B. PCI-Bus, Serielle Busse wie z. B. Ethernet)
c) Kopplung an einen internen Registersatz oder mehrere interne Registersätze
d) andere Speichermedien (Festplatte, Flash-ROM, etc.)

[0013] Es sei erwähnt, daß hier als ein "Prozessor" auch ein Dual-Prozessorsystem und/oder ein andere Anordnungen aufweisendes System insbesondere mit mehr als zwei konventionellen Prozessoren gemeint sein kann. Gleichfalls und/oder alternativ sind mehrere rekonfigurierbare Bausteine gleichzeitig verwendbar.

Übersetzungsprinzip

[0014] Aus einem PROGRAMM werden mittels eines PRÄPROZESSORS die Teile extrahiert, die sich auf die jeweils bestimmte(n) VPU(s) effizient und/oder sinnvoll abbilden lassen. Diese Teile werden in einem für VPUs geeigneten Format ausgegeben (NML).

[0015] Der verbleibende Code und/oder der extrahierte Code wird an der Stelle der durch die Extraktion fehlenden Code-Teile um einen Interface-Code erweitert, der entsprechend der Architektur des Zielsystems die Kommunikation zwischen PROZESSOR(en) und VPU(s) steuert. Der verbleibende und ggf. erweiterte Code wird in Form einer klassischen Hochsprache ausgegeben (HOSTCODE), wobei die Gattung des ausgegebenen Codes insbesondere exakt der Gattung der ursprünglichen HOCHSPRACHE entsprechen kann.

[0016] Diese extrahierte HOCHSPRACHE wird mittels eines gewöhnlichen Standard-Übersetzers für den/die jeweiligen PROZESSOR(en) übersetzt, wobei es möglich ist, dies so zu gestalten, daß keinerlei besondere Anpassung des Übersetzers an die verwendete Datenverarbeitungsarchitektur notwendig ist.

[0017] Durch dieses Verfahren wird der Implementierungsaufwand der Programmierumgebung erheblich vereinfacht. Zudem kann der Anwender den PROZESSOR weiterhin in der ihm bekannten Programmierumgebung, die er frei wählen kann, programmieren und debuggen.

Übersetzungsablauf

Extraktion

[0018] Zunächst wird der für eine VPU geeignet erscheinende Code aus dem PROGRAMM extrahiert. Die Extraktion kann auf unterschiedlichen Methoden basieren, die einzeln oder kombiniert angewendet werden. Folgende Methoden sollen beispielhaft detaillierter beschrieben werden.

Extraktion durch Hints

[0019] Der Programmierer gibt explizit durch Hinweise (Hints) innerhalb des PROGRAMMES Anweisungen, welche Teile extrahiert werden sollen. Beispielsweise kann dies

folgendermaßen erfolgen:

```
...
Code
...
//START_NML_EXTRACTION
Zu extrahierender Code
//END_NML_EXTRACTION
...
Code
...
```

"//START_NML_EXTRACTION" kennzeichnet den Beginn eines zu extrahierenden Codes.
"//END_NML_EXTRACTION" kennzeichnet das Ende eines zu extrahierenden Codes.

[0020] In einem solchen Fall ist die Einheit zur Umsetzung des Programms in Konfigurationscodes dazu ausgebildet, die Hints beziehungsweise Umsetzungsvorgaben zu erkennen.

Extraktion durch Aufruf von NML-Routinen

[0021] Der Programmierer implementiert Teile des PROGRAMMES direkt in NML und springt in die NML-Routinen durch Aufrufe (calls). Beispielsweise erfolgt dies derart:

```
a) NML-Code
...
procedure EXAMPLE
begin
...
end
...
b) PROGRAMM Code
...
Code
...
call EXAMPLE // Aufruf des NML-Codes
...
Code
...
```

[0022] In diesem Fall ist die Einheit zur Umsetzung dazu ausgebildet, NML-Programmteile, das heißt Programmteile zur Ausführung in und/oder auf einem rekonfigurierbaren Array in ein größeres Programm einzubinden.

Extraktion durch Analyse

[0023] Durch an die jeweilig VPU angepaßte Analysemethoden werden Teile innerhalb des PROGRAMMES erkannt, die effizient und/oder sinnvoll auf die VPU abbildbar sind. Diese Teile werden aus dem PROGRAMM extrahiert.

[0024] Eine beispielsweise für viele VPUs geeignete Analysemethode ist der Aufbau von Datenfluß- und/oder Kontrollflußgraphen aus dem PROGRAMM. Diese Graphen können hinsichtlich ihrer möglichen Partitionierung und/oder Abbildung auf die Ziel-VPU automatisch untersucht werden. In diesem Fall werden die Teile der generierten Graphen bzw. die entsprechenden PROGRAMM-Teile, extrahiert; die sich hinreichend gut partitionieren und/oder abbilden lassen. Hierzu kann eine Partitionierbarkeits- und/oder Abbildbarkeitsanalyse erfolgen, die die jeweilige Eigenschaft bewertet.

Übersetzung in NML

[0025] Eine für die implementierte VPU geeignete Übersetzung des extrahierten Codes nach NML wird durchgeführt.

[0026] Für datenflußorientierte VPUs kann beispielsweise automatisch ein Datenfluß- und/oder Kontrollflußgraph aufgebaut werden. Die Graphen werden dann in NML-Code übersetzt. Dabei kann gegebenenfalls bereits das Abbilden auf die VPU erfolgen, beispielsweise mittels der Durchführung des Plazierens der benötigten Ressourcen und des Routens der Verbindungen (Place and Route). Dies geschieht zum Beispiel nach typischen bekannten Regeln des Plazierens und Routens.

Analyse

[0027] Mittels einer automatischen Analysemethode wird der extrahierte Code und/oder der übersetzte NML-Code auf seine Verarbeitungseffizienz hin analysiert. Dabei ist die Analysemethode bevorzugt so gewählt, daß der Interface-Code und die daraus entstehenden Performanceeinflüsse an geeigneter Stelle mit in die Analyse einfließen.

[0028] Gegebenenfalls wird die Analyse durch eine komplette Übersetzung und Implementierung auf dem Hardware-System durchgeführt, indem das PROGRAMM ausgeführt und mit geeigneten Methoden, wie sie beispielsweise nach dem Stand der Technik bekannt sind, vermessen wird.

Loop

[0029] Basierend auf den durchgeführten Analysen, können verschiedene durch die Extraktion für eine VPU gewählte Teile als ungeeignet identifiziert werden. Umgekehrt kann die Analyse ergeben, daß bestimmte für einen PROZESSOR extrahierte Teile zur Ausführung auf einer VPU geeignet wären.

[0030] Eine optionale Schleife, die nach der Analyse basierend auf geeigneten Entscheidungskriterien zurück in den Extraktionsteil führt, um diesen mit entsprechend der Analyse angepaßten Extraktionsvorgaben erneut auszuführen, ermöglicht die Optimierung des Übersetzungsergebnisses. Man hat somit eine Iteration.

[0031] Die Schleife kann an mehreren unterschiedlichen Stellen in den Compilerlauf eingebracht sein.

Einbindung der PROZESSOR- und VPU-Compiler

[0032] Der ausgegebene Code ist üblicherweise vollständig und ohne weitere Eingriffe auf den jeweils nachfolgenden Compilern ausführbar. Gegebenenfalls werden Compilerflags und Constraints für die nachfolgenden Compiler generiert, wobei der Anwender optional eigene Vorgaben hinzufügen und/oder die generierten Vorgaben modifizieren kann. Die nachfolgenden Compiler benötigen keine wesentlichen Modifikationen, sodaß Standard-Tools einsetzbar sind.

[0033] Das vorgeschlagene Verfahren eignet sich somit beispielsweise insbesondere als Präprozessor vor Compilern und Entwicklungssystemen.

Interface-Code

[0034] Der Interface-Code, der in den extrahierten Code eingesetzt wird, kann durch unterschiedliche Verfahren vorgegeben werden. Bevorzugt wird der Interface-Code in einer Datenbank abgelegt, auf die zugegriffen wird. Die Einheit zur Umsetzung kann so ausgebildet sein, daß sie eine

Auswahl des Programmierers berücksichtigt, der beispielsweise durch Hinweise im PROGRAMM oder durch Compilerflags den passenden Interface-Code auswählt. Dabei kann der für das jeweils verwendete Implementierungsverfahren geeignete Interface-Code gewählt werden.

[0035] Die Datenbank selbst kann durch unterschiedliche Methoden aufgebaut und gewartet werden. Einige Beispiele sollen zur Verdeutlichung der Möglichkeiten angeführt werden:

a) Der Interface-Code kann vom Lieferanten des Compilers für bestimmte Verbindungsverfahren vorgegeben werden. Dies kann bei der Organisation der Datenbank berücksichtigt werden, indem entsprechende Speichermittel für diese Angaben bereitgehalten werden.

b) Der Interface-Code kann vom Benutzer, der den Systemaufbau bestimmt hat, selbst geschrieben oder aus bestehenden (Beispiel-) Interface-Codes modifiziert und der Datenbank zugefügt werden. Das Datenbankmittel wird hierzu bevorzugt benutzermodifizierbar gestaltet, um dem Benutzer die Datenbankmodifikation zu ermöglichen.

c) Der Interface-Code kann von einem Entwicklungssystem, mit dem beispielsweise der Systemaufbau geplant und/oder beschrieben und/oder getestet wurde, automatisch generiert werden.

[0036] Der Interface-Code ist gewöhnlicherweise derart gestaltet, daß er den Anforderungen der Programmiersprache entspricht, in der der extrahierte Code vorliegt in den der Interface-Code eingefügt werden soll.

Debugging und Integration der Toolsets

[0037] In die Interface-Codes können Kommunikationsroutinen eingeführt werden, um die unterschiedlichen Entwicklungssysteme für PROZESSOR und VPU zu synchronisieren. Insbesondere kann der Code für die jeweiligen Debugger aufgenommen werden.

[0038] Der Interface-Code steuert den Datenaustausch zwischen PROZESSOR und VPU. Er ist daher eine geeignete und bevorzugte Schnittstelle, um die jeweiligen Entwicklungssysteme und Debuggets zu steuern. Es ist beispielsweise möglich, einen Debugger für den PROZESSOR solange zu aktivieren, wie die Daten von dem Prozessor verarbeitet werden. Sobald die Daten über den Interface-Code an eine (oder mehrere) VPU übergeben werden, ist ein Debugger für VPUs zu aktivieren. Wird der Code zurück an den PROZESSOR gesendet, soll wiederum der PROZESSOR-Debugger aktiviert werden.

[0039] Es ist daher also möglich und bevorzugt, derartige Abläufe durch das Einfügen von Steuercodes für Debugger und/oder Entwicklungssysteme in den Interface-Code abzuwickeln.

[0040] Die Kommunikation und Steuerung zwischen den unterschiedlichen Entwicklungssystemen soll daher bevorzugt mittels in die Interface-Codes von PROZESSOR und/oder VPU eingebrachte Steuercodes abgewickelt werden. Die Steuercodes können dabei bestehenden Standards für die Steuerung von Entwicklungssystemen weitgehend entsprechen.

[0041] Die Verwaltung und Kommunikation der Entwicklungssysteme wird vorzugsweise wie beschrieben in die Interface-Codes abgewickelt, kann jedoch – sofern sinnvoll – auch getrennt von diesen, nach einem entsprechenden ähnlichen Verfahren abgewickelt werden.

Beschreibung der Figuren

[0042] **Fig. 1** verdeutlicht das vorgeschlagene Verfahren und zeigt einen möglichen Systemaufbau. Dabei ist ein PROZESSOR (**0101**) über ein geeignetes Interface (**0102**) zum Daten- und Statusaustausch mit einer VPU (**0103**) verbunden.

[0043] Ein PROGRAMM-Code (**0110**) wird in einen für den PROZESSOR geeigneten Teil (**0111**) und einen VPU-geeigneten Teil (**0112**) zerlegt.

[0044] **0111** wird durch einen Standard-Compiler (**0113**) übersetzt, wobei zusätzlicher Code zur Beschreibung und Verwaltung des Interfaces (**0102**) aus einer Datenbank (**0114**) eingefügt wird. Auf **0101** ausführbarer sequentieller Code wird generiert (**0116**) und sofern notwendig die entsprechende Programmierung (**0117**) des Interfaces (**0102**).

[0045] **0112** wird durch einen VPU-Compiler (**0115**) übersetzt, wobei ein zusätzlicher Code zur Beschreibung und Verwaltung des Interfaces (**0102**) aus einer Datenbank (**0114**) eingefügt wird. Auf **0103** ausführbare Konfigurationen werden generiert (**0118**) und sofern notwendig die entsprechende Programmierung (**0119**) des Interfaces (**0102**).

[0046] In **Fig. 2** ist beispielhaft ein prinzipieller Ablauf einer Compilation dargestellt. Ein PROGRAMM (**0201**) wird in der Extraktionseinheit (**0202**) nach unterschiedlichen Verfahren in VPU-Code (**0203**) und PROZESSOR-Code (**0204**) zerlegt. Unterschiedliche Methoden können in beliebiger Kombination zur Extraktion angewendet werden, beispielsweise Hinweise im ursprünglichen PROGRAMM (**0205**), Unterprogrammaufrufe (**0206**) und/oder Analyseverfahren (**0207**). Der jeweils extrahierte Code wird ggf. übersetzt und ggf. auf seine Eignung für das jeweilige Zielsystem hin überprüft (**0208**). Dabei ist eine Rückkopplung (**0209**) auf die Extraktion möglich, um Verbesserungen durch eine geänderte Zuordnung der Codes zum PROZESSOR oder einer VPU zu erhalten.

[0047] Danach (**0211**) wird **0203** durch den Interface-Code aus einer Datenbank (**0210**) erweitert (**0212**) und/oder **0204** wird durch den Interface-Code aus **0210** zu **0213** erweitert.

[0048] Der entstandene Code wird auf seine Performance analysiert (**0214**), ggf. ist eine Rückkopplung (**0215**) auf die Extraktion möglich, um Verbesserungen durch eine geänderte Zuordnung der Codes zum PROZESSOR oder einer VPU zu erhalten.

[0049] Der entstandene VPU-Code (**0216**) wird für eine weitere Übersetzung an einen nachgeschalteten für die VPU geeigneten Compiler weitergegeben. Der entstandene PROZESSOR-Code (**0217**) wird für die weitere Übersetzung in einem beliebigen nachgeschalteten für den PROZESSOR geeigneten Compiler weiterverarbeitet.

[0050] Es soll angemerkt werden, daß einzelne Schritte je nach Verfahren ausgelassen werden können. Wesentlich ist, daß weitgehend kompletter und ohne Eingriff durch den Programmierer direkt übersetzbarer Code an die jeweils nachgeschalteten Compilersysteme ausgegeben wird.

[0051] Die Datenbank für die Interface-Codes (**0210**) wird unabhängig und vor dem Compilerdurchlauf aufgebaut. Beispielsweise sind folgende Quellen für die Datenbank möglich: Vom Lieferanten vorgegeben (**0220**), vom Benutzer programmiert (**0221**) oder automatisch von einem Entwicklungssystem generiert (**0222**).

[0052] Zusammenfassend befaßt sich die vorliegende Erfindung mit Verfahren, die eine Übersetzung einer klassischen Hochsprache wie Pascal, C, C++, Java, etc. auf eine rekonfigurierbare Architektur ermöglicht. Das Verfahren ist derart ausgelegt, daß nur die jeweils für die rekonfigurierbare Zielarchitektur geeigneten Teile des zu übersetzenden

Programmes extrahiert werden. Die verbleibenden Teile des Programmes werden auf eine konventionelle Prozessorarchitektur übersetzt.

Patentansprüche

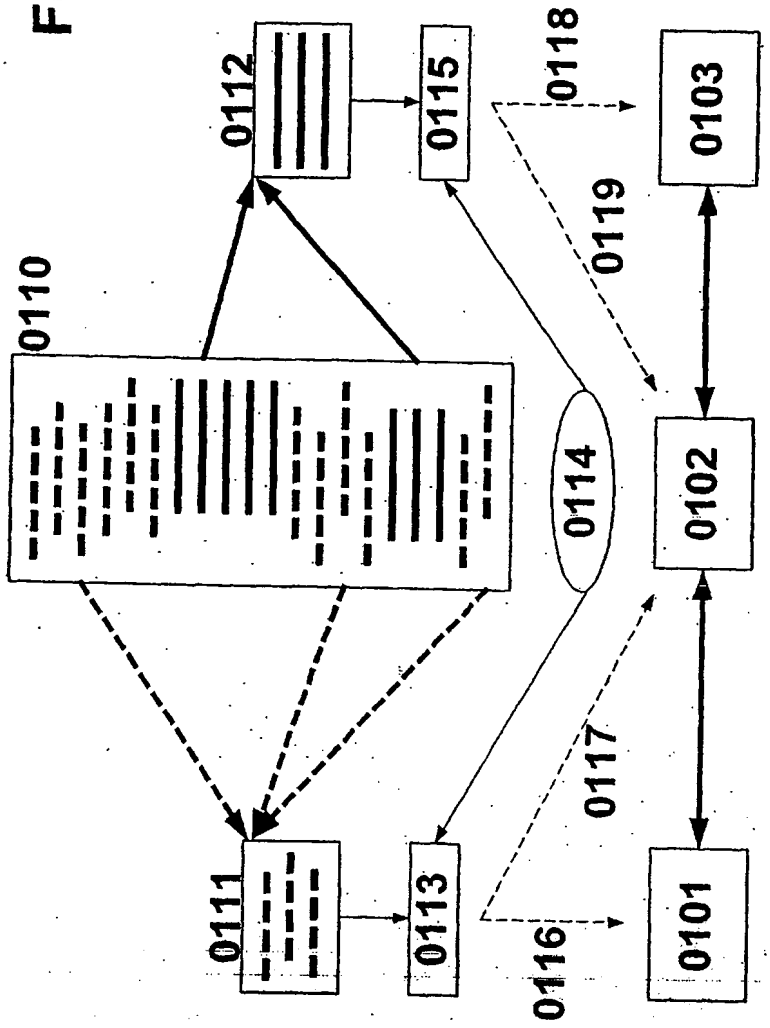
1. Verfahren zur Übersetzung von Programmen auf ein System bestehend aus wenigstens einem ersten Prozessor und einer rekonfigurierbaren Einheit, **dadurch gekennzeichnet**, daß die Codeteile, die für die rekonfigurierbare Einheit geeignet sind, bestimmt und extrahiert werden und der verbleibende Code zur Abarbeitung durch den ersten Prozessor derart extrahiert wird.
2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß dem für den Prozessor extrahierten Code derart Interface-Code zugefügt wird, daß eine Kommunikation zwischen Prozessor und rekonfigurierbarer Einheit entsprechend des Systems möglich ist.
3. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß dem für die rekonfigurierbare Einheit extrahierten Code derart Interface-Code zugefügt wird, daß eine Kommunikation zwischen Prozessor und rekonfigurierbarer Einheit entsprechend des Systems möglich ist.
4. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß der zu extrahierende Code aufgrund von Analysen festgestellt wird.
5. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß der zu extrahierende Code aufgrund von Hinweisen im Code festgestellt wird.
6. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß der zu extrahierende Code aufgrund von Aufrufen von Unterprogrammen festgestellt wird.
7. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß der Interface-Code eine Speicherkopplung (Shared-Memory) vorsieht.
8. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß der Interface-Code eine Registerkopplung vorsieht.
9. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß der Interface-Code eine Kopplung mittels eines Netzwerkes vorsieht.
10. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß der extrahierte Code analysiert wird und gegebenenfalls die Extraktion mit neuen verbesserten Parametern erneut gestartet wird.
11. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß dem extrahierten Code Steuer-Code zur Verwaltung und/oder Steuerung und/oder Kommunikation der Entwicklungssysteme zugefügt wird.
12. Verfahren nach einem der vorhergehenden Ansprüche, worin der erste Prozessor eine konventionelle Prozessorarchitektur aufweist, insbesondere ein Prozessor mit von -Neumann- und/oder Harvardarchitektur, Kontroller, CISC-, RISC-, VLIW-, DSP-Prozessor.
13. Verfahren insbesondere nach einem der vorhergehenden Ansprüche zur Übersetzung von Programmen auf ein System bestehend aus einem Prozessor und einer rekonfigurierbaren Einheit, dadurch gekennzeichnet, daß
die Codeteile, die für die rekonfigurierbare Einheit geeignet sind, extrahiert werden,
der verbleibende Code derart extrahiert wird, daß er mittels eines beliebigen gewöhnlichen unmodifizierten für den Prozessor geeigneten Compilers übersetzbar ist.
14. Vorrichtung zur Datenverarbeitung mit wenigstens einem herkömmlichen Prozessor und wenigstens einer rekonfigurierbaren Einheit, dadurch gekennzeichnet, daß sie ein Mittel zum Informationsaustausch, insbesondere in Form von Daten- und Statusinformation zwischen herkömmlichem Prozessor und rekonfigurierbarer Einheit aufweist, wobei das Mittel so ausgebildet ist, daß eine Daten- und Statusinformation zwischen denselben während der Abarbeitung eines oder mehrerer Programme möglich ist und/oder ohne daß insbesondere die Datenverarbeitung auf dem rekonfigurierbaren Prozessor und/oder dem herkömmlichen Prozessor signifikant unterbrochen werden muß.

Hierzu 2 Seite(n) Zeichnungen

- Leerseite -

Nummer: DE 101 29 237 A1
Int. Cl.[7]: G 06 F 9/45
Offenlegungstag: 18. April 2002

Fig. 1



Nummer: DE 101 29 237 A1
Int. Cl.[7]: G 06 F 9/45
Offenlegungstag: 18. April 2002

Fig. 2